INTERNSHIP AT ASIMOV ROBOTICS

Internship Report

Submitted to the APJ Abdul Kalam Technological University

in partial fulfillment of requirements for the award of degree

Master of Technology

in

Robotics and Automation

by

MUNAVIR ZAMAN P K(TVE22ECRA13)

कर्म ज्यायोध्य कर्मण

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING TRIVANDRUM

KERALA

MAY - JUNE 2023

DECLARATION

I, Munavir Zaman P K declare that this Internship Report titled internship at ASIMOV ROBOTICS represents my own original work conducted during my internship at ASIMOV ROBOTICS. The contents of this report are based on my personal experiences, observations, and interactions within the organization.

I confirm that this report has not been submitted for any other academic purpose or assessment. All sources of information and data used in this report have been duly acknowledged and referenced.

I understand the strict policy against plagiarism and academic misconduct, and I affirm that this report is free from any form of plagiarism. The opinions, findings, and conclusions presented in this report are solely my own and do not necessarily reflect those of ASIMOV ROBOTICS Limited or any other entity.

I would like to express my heartfelt appreciation to the individuals at ASIMOV ROBOTICS who provided guidance, support, and valuable insights during my internship. Their contributions have greatly enhanced my learning experience and professional growth.

I hereby submit this report as a comprehensive record of my internship at ASIMOV ROBOTICS and to fulfill the requirements of the internship program.

MUNAVIR ZAMAN P K

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Mr.Raghavendra B M, my internship supervisor, for providing me with the opportunity to work on this project and for their guidance and support throughout the internship. Their expertise and valuable insights greatly contributed to the successful completion of the 3R planar manipulator.

I would also like to thank the entire team at ASIMOV Robotics for their collaboration and assistance during the internship. Their willingness to share knowledge, answer questions, and offer assistsance whenever needed played a significant role in the project's progress. Additionally, I extend my thanks to Afna Shereef and Uthraj M for their collaboration and teamwork. Their contributions and dedication were instrumental in the successful implementation of the docking system.

I acknowledge that any errors or omissions in this report are entirely my own.

MUNAVIR ZAMAN P K

Abstract

This project focuses on the development of a docking system using a 3-degree-of-freedom planar robotic manipulator. The objective is to design and control a robot arm capable of accurately positioning and aligning with target objects. The project encompasses various stages, including simulation, motion planning, hardware implementation, and performance evaluation. In the simulation phase, the ROS framework, along with RViz and Gazebo, is utilized to create a realistic environment for visualizing and studying the behavior and kinematics of the RRR robot model. The URDF model is then created, defining the structure, dimensions, and joint properties of the robot arm. The MoveIt package is developed for motion planning and control, incorporating trajectory planning and inverse kinematics algorithms. The project also involves interfacing and testing hardware components, such as workspace point identification, object position detection, camera calibration, and dynamixel motors. The physical movements of the robot arm are validated against the simulated results, ensuring consistency and accuracy. Performance evaluation is conducted by comparing simulated and hardware performance. Key findings from the project include the suitability of a 3-degree-of- freedom planar manipulator for docking applications and the importance of simulation in system development. Insights are gained regarding motion planning, inverse kinematics, and hardware calibration. The developed docking system has potential applications in space exploration, manufacturing, and logistics. Future work can focus on further enhancing the system's performance, integrating advanced sensing technologies, exploring advanced control strategies, and optimizing it for specific docking scenarios.

Contents

Chapter 1

Introduction

The field of robotics has witnessed significant advancements in recent years, with applications ranging from manufacturing to space exploration. One crucial aspect of robotic systems is their ability to perform precise and controlled movements. In the context of docking operations, where a robotic arm needs to align and connect with a target object, accurate positioning is importance. This project focuses on developing a docking system using a 3-degree-of-freedom planar robotic manipulator, leveraging the capabilities of ROS (Robot Operating System) and the MoveIt framework. Docking operations play a crucial role in various industries, including aerospace, marine, and manufacturing. Efficient and reliable docking systems are essential for tasks such as satellite servicing, assembly of modular structures, and automated material handling. The use of robotic manipulators in these operations offers numerous advantages, including increased precision, flexibility, and repeatability. The motivation behind this project is to design and implement a docking system that can accurately position an end effector to a target position, providing a reliable and efficient solution for docking tasks.

Chapter 2

Objectives

The goal of this project is to create a docking system that can successfully place an end effector in a target position using a 3R planar manipulator. The project calls for the use of ROS, the development of a URDF model, the use of motion planning algorithms, and object detection and position calculation with the help of camera and the interaction of real-time control hardware with software components.

- Develop a URDF (Unified Robot Description Format) model that accurately represents the physical structure, dimensions, joint properties of the robotic arm and visualize it in a simulated environment.
- Utilize the MoveIt framework to implement motion planning and control for the manipulator.
- Study trajectory planning techniques and implement inverse kinematics algorithms to compute joint angles for precise positioning of the end effector.
- Inteface with hardware components, including sensors, cameras, and motors, to bridge the gap between simulation and real-world operation.
- Calibrate the chosen camera and do the object detection and position estimation task with the help of appropriate technology.
- Evaluate the performance of the arm and analyze the achieved accuracy and reliability in positioning the end effector.

Chapter 3

Importing the package/URDF generated from SOLIDWORKS in ROS

3.1 ROS installation

ROS (Robot Operating System) is a flexible framework for writing robot software. ROS Noetic is one of the major releases of ROS, and it is compatible with Ubuntu 20.04 (Focal Fossa).

3.2 Setup a CATKIN Workspace

Setting up a catkin workspace is essential for developing and building ROS packages.

3.3 Copy the generated Package to catkin workspace

The first step is to Copy the ROS package created using Solidworks in the “src” folder of catkin workspace .

3.3.1 Folder and files available in the exported package

1. config: This folder contains a yaml file containing names of joints available in URDF file which is generated.

2. launch: This folder contains two files. “display.launch” which is used to open the robot in Rviz. “gazebo.launch” is used to launch robot in GAZEBO.

3. meshes: This folder contains “.stl” files of all the links in your robot.

4. textures: this folder will be empty.

5. URDF:This folder contains a URDF file of your Robot. This file is a description of you robot containing information of links, joints, positions and ranges of motion of joints, their mass properties, inertial properties etc. Also, this folder contains a your_package_name.csv file containing information of your robot’s links.

6. CMakeLists.txt: This file is the input to the CMake build system for building software packages. It describes how to build the code and where to install it. You should not rename or change the sequence of code in this file

7. export: Contains logs generated while creating the package in SOLIDWORKS

8. package.xml: This file defines properties about the package such as the package name, version numbers, authors, maintainers, and dependencies on other catkin packages. system package dependencies are declared in package.xml. If they are missing or incorrect our package may or may not work.

Chapter 4

Control the Robotic Arm URDF using Moveit

Moveit is a great open source Robot manipulation tool which can be used to create packages, which can be used to manipulate any Robot in ROS.

4.1 Creating the Moveit Package using Moveit Setup Assistant to control our URDF file.

Creating a MoveIt package using the MoveIt Setup Assistant is a critical step in setting up robotic systems within the Robot Operating System (ROS) framework. This package serves as a bridge between your robot's hardware and the software tools required for motion planning and control. It allows you to define your robot's kinematic and dynamic properties, specify its planning groups and end-effectors, and configure its collision model, among other essential parameters. By creating and configuring this package, you streamline the integration of advanced robotic capabilities like motion planning, collision detection, and trajectory execution into your project. It provides a standardized and efficient way to control your robot, making it easier to develop and deploy applications, ranging from autonomous navigation to robotic manipulation, all while benefiting from the extensive ROS ecosystem and community support.

4.2 Forward kinematics

Forward kinematics is a process in robotics that involves determining the position and orientation of the end effector (e.g., gripper or tool) of a robot manipulator based on the joint angles or joint positions of the robot's joints. It helps in understanding the robot's configuration in the workspace and enables tasks such as motion planning, trajectory generation, and control.The Denavit-Hartenberg (DH) convention is a widely used method for representing the kinematics of serial-link robotic manipulators. It provides a systematic way to define the relationship between adjacent links and joints using a set of parameters.In the forward kinematics process, the DH parameters, which include joint angles, link lengths, link offsets, and link twists, are used to construct a series of transformation matrices that represent the transformation from one joint to the next. These transformation matrices are multiplied together to obtain the transformation matrix that represents the overall position and orientation of the end effector with respect to a reference frame (typically the base frame).The transformation matrices are computed based on the DH parameters using trigonometric functions and basic matrix operations. Each transformation matrix captures the rotation and translation between two adjacent joints or links. By multiplying these matrices, we obtain the transformation matrix that represents the complete kinematic chain of the robot arm.Once the transformation matrix for the end effector is obtained, the position and orientation of the end effector can be extracted from the matrix. The position represents the Cartesian coordinates (x, y, z) of the end effector in the reference frame, and the orientation is typically represented using Euler angles or quaternions.By performing forward kinematics, we can determine the pose of the end effector given the joint angles or joint positions. This information is essential for various robot-related tasks, such as path planning, obstacle avoidance, and manipulation tasks.

4.3 Inverse kinematics

Inverse kinematics is a fundamental concept in robotics that deals with the calculation of joint angles required to position a robotic arm's end effector (e.g., gripper or tool) at a desired target position and orientation in space. It is an essential aspect of motion planning and control for robot arms, as it enables precise control of the arm's movement to reach specific points in its workspace.The process of inverse kinematics involves working backwards from the desired end effector pose to calculate the joint angles required to achieve that pose. It is in contrast to forward kinematics, where given the joint angles, the end effector's position and orientation are computed.In practical terms, robotic manipulators often have multiple degrees of freedom (DOF) represented by different joints (e.g., revolute, prismatic, etc.). Solving inverse kinematics analytically can be complex or even impossible for some robot configurations. However, numerical methods, optimization techniques, and numerical solvers can be used to approximate the solutions effectively.In the context of the Robot Operating System (ROS), the MoveIt package is a powerful and widely used motion planning framework. It provides various tools and libraries to simplify the process of performing inverse kinematics calculations for robot arms. MoveIt offers different algorithms and solvers for solving inverse kinematics problems for various robot types and configurations.By utilizing MoveIt's inverse kinematics capabilities, we can plan and execute precise and safe trajectories for their robotic arms, making it easier to interact with objects and perform complex tasks in real-world applications such as manufacturing, research, logistics, and more.

4.4 Simulation in Rviz and Gazebo

Simulation of a robotic arm in Rviz and Gazebo provides essential tools for robotics development and research. These tools are widely used within the ROS (Robot Operating System) ecosystem and offer complementary features that enhance the simulation experience.

Rviz:

Rviz (ROS Visualization) is a 3D visualization tool in ROS, enabling the visualization of robot models, sensor data, and other information from the ROS system. To simulate a robotic arm in Rviz, you need the robot's URDF (Unified Robot Description Format), which defines its physical structure, and joint states published as ROS topics. Rviz offers interactive control of the robot's joints through a user-friendly GUI, allowing users to visualize how the arm moves in response to different joint configurations.

Additionally, Rviz can display sensor data, facilitating the testing of perception algorithms and verifying the arm's interactions with the environment.

Gazebo:

Gazebo is a powerful physics simulator used to create dynamic and realistic simulations of robotic systems. Simulating a robotic arm in Gazebo requires the robot's URDF, joint states, and a controller or Gazebo plugin to apply joint efforts and simulate the arm's motion. Gazebo provides realistic dynamics, including gravity, friction, and joint constraints, making it suitable for testing and tuning control algorithms. It also supports collision detection using the robot's collision model defined in the URDF, enabling physics-based interactions with the environment. Gazebo can simulate sensor data, such as camera images or laser scans, which are valuable for testing perception and mapping algorithms.

Integration of Rviz and Gazebo:

Combining Rviz and Gazebo creates a comprehensive simulation environment for robotic arms. The simulated robot model from Gazebo can be visualized in Rviz, enhancing visualization and debugging capabilities for the robot's kinematics. Joint states and sensor data from Gazebo can be visualized in Rviz, providing a complete view of the robot's state and its perception of the environment.This simulation approach is instrumental in designing, testing, and optimizing robotic arms, enabling researchers to refine control algorithms and assess the robot's performance in a virtual environment before deploying it on a physical system.

In this project conducted a comprehensive evaluation of the robot arm's performance using the MoveIt package and an implemented inverse kinematics algorithm. The primary objective was to assess the arm's ability to move to desired end-effector positions accurately and efficiently.To achieve this,defined the robot's kinematic properties and configured the MoveIt package to suit our specific robot model. We also set clear target end-effector positions and orientations for the arm to reach. The implemented inverse kinematics code played a pivotal role in computing the necessary joint configurations to achieve these targets. Utilizing the MoveIt RViz interface,interactively planned and executed arm movements, scrutinizing the system's responsiveness and precision.

Chapter 5

Camera Calibration and Object Detection

Camera calibration and object detection are fundamental techniques in computer vision and robotics. Camera calibration ensures that images captured by a camera accurately represent the real world by correcting for lens distortion and providing essential parameters for mapping 3D objects into 2D images. This process is vital for tasks like 3D reconstruction, accurate measurements, and robot perception. On the other hand, object detection involves the use of machine learning models to identify and locate objects within images or video frames. It enables automation in various domains, including autonomous vehicles, surveillance, manufacturing, and augmented reality, by allowing machines to recognize and interact with objects in their environment. Combining camera calibration with object detection enhances the capabilities of systems, providing accurate visual perception for a wide range of applications, from industrial automation to cutting-edge technological innovations.

5.1 Visp

Visp standing for Visual Servoing Platform is a modular cross platform library that allows prototyping and developing applications using visual tracking and visual servoing techniques at the heart of the research done by IRISA - Inria Rainbow team (previously Lagadic team). ViSP is able to compute control laws that can be applied to robotic systems. It provides a set of visual features that can be tracked using real time image processing or computer vision algorithms. ViSP also provides simulation capabilities. ViSP can be useful in robotics, computer vision, augmented reality and computer animation.

5.2 OpenCV

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimised library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.

For the accurate solution ,first we have to do camera calibration with openCV .The process of estimating the parameters of a camera is called camera calibration.This means we have all the information (parameters or coefficients) about the camera required to determine an accurate relationship between a 3D point in the real world and its corresponding 2D projection (pixel) in the image captured by that calibrated camera.
Typically this means recovering two kinds of parameters.

1 Internal parameters of the camera/lens system. E.g. focal length, opticalcenter, and radial distortion coefficients of the lens.

2 External parameters : This refers to the orientation (rotation and translation) of the camera with respect to some world coordinate system. We use a chessboard for the camera calibration task.Because checkerboard patterns are distinct and easy to detect in an image. Not only that, the corners of squares on the checkerboard are ideal for localising them because they have sharp gradients in two directions.

Chapter 6

Interface With Hardware

Integrating hardware with a ROS simulation using MoveIt entails configuring physical robotic components, developing ROS nodes for hardware communication, modeling the robot in URDF, and incorporating MoveIt for motion planning and control. This comprehensive process enables the creation of a unified environment where the simulated robot interacts with the physical hardware, allowing for testing and development in a safe and controlled manner, ultimately enhancing the efficiency and reliability of robotic applications.

In the context of hardware interfacing, the focus lies in establishing seamless communication between the ROS simulation and the physical robotic components. The first crucial step involves creating a Unified Robot Description Format (URDF) model that accurately represents the robot's kinematics, dynamics, and geometry. This model serves as the foundation upon which the entire integration is built, ensuring that the simulation closely mirrors the real-world behavior of the hardware.

The next key element is the development of ROS nodes tailored for hardware communication. These nodes act as intermediaries, receiving commands and data from the ROS environment, including motion plans generated by MoveIt, and translating these instructions into actions for the physical hardware. Whether it's controlling robotic arms, sensors, or actuators, these nodes play a pivotal role in bridging the virtual and physical realms, allowing for real-time interaction and control.

Finally, the incorporation of MoveIt into the hardware interface adds an essential layer of intelligence. MoveIt facilitates motion planning, collision avoidance, and trajectory generation, ensuring that the robotic hardware can perform complex tasks accurately and safely. By combining these components, the hardware interface with ROS and MoveIt creates a powerful platform for robotics development, enabling engineers to refine and test algorithms, control strategies, and applications in a simulated environment before deploying them on physical hardware, thus minimizing risks and optimizing the overall development process. The robotic arm comprises four Dynamixel motors, renowned for their robustness and versatility in the field of robotics. These motors serve as the primary actuators responsible for driving the arm's movements and articulations. To facilitate seamless control and integration, we have chosen to interface these motors with the U2D2 communication interface, a reliable and widely adopted solution. This interface establishes bidirectional communication channels, allowing data and commands to flow between the robotic arm and the controlling software, be it for trajectory planning, real-time adjustments, or feedback monitoring. This integration forms the backbone of our hardware control system, enabling precise and responsive control over the robotic arm's movements within the ROS simulation and real-world environments.

6.0.1 Hardware Components

1 Robotic Arm:It is a mechanical device, It is typically composed of several interconnected components that work together to achieve precise and controlled movements.The main components are base,joints,links andmotors.Here the arm has 4 links base link,link 1,link 2,link 3 respectively and 3 revolute joints joint 1,joint 2,joint 3 respectively.

2 U2D2 Communication Interface: Is a communication device used in robotics to establish a connection between controlling software and Dynamixel motors.

3 Power Supply

4 Wiring and Connections

5 Dynamixel MX-28 Motors: The Dynamixel MX-28 is a high-performance servo motor designed for use in robotics and mechatronics applications. Manufactured

by ROBOTIS, the MX-28 motor is known for its precision, power, and versatility, making it a popular choice among roboticists.

Key Features of Dynamixel MX-28 Motors:

- High Torque and Power: The MX-28 motor offers impressive torque capabilities, allowing it to generate substantial force for various robotic tasks.
- With a resolution of 0.088 degrees, the MX-28 provides precise control over joint movements, enabling accurate positioning and smooth motion.
- 360-Degree Rotation: The motor offers continuous rotation, allowing for unlimited movement in both directions.
- Dual Ball Bearings: Dual ball bearings ensure smooth operation and enhance the motor's durability and lifespan.
- Robust Construction: The MX-28 motor features a robust build quality, ensuring reliable performance even in demanding environments.
- Advanced Control Features: The motor supports various control modes, including position control, velocity control, and torque control, providing flexibility in control strategies.
- Communication Protocol : The MX-28 motor utilizes the TTL communication protocol for bidirectional communication with the controlling hardware. This protocol enables real-time data transmission between the motor and the controller, allowing for precise control and monitoring of the motor's status.
- Feedback System : The motor incorporates a built-in absolute encoder, which provides accurate feedback on the motor's position, velocity, and temperature. This feedback enables closed-loop control and enhances the motor's ability to maintain precise positions and execute complex movements.
- Software Support : ROBOTIS provides comprehensive software support for the MX-28 motor, including development libraries, APIs, and programming examples. These resources facilitate integration with popular robotics platforms and simplify the implementation of control algorithms.

6.0.2 Verify the hardware performance

Precise motor control has been achieved through the application of forward and inverse kinematics, enabling accurate positioning of the arm based on joint angles and end effector positions. In the verification process, we conduct a crucial step by comparing the actual joint state values of the Dynamixel motors with the joint state values in the simulation. This comparative analysis serves as a robust method to ensure that the physical robotic arm moves to the desired positions accurately and in alignment with the simulated trajectories, confirming the reliability and accuracy of the hardware control and the seamless integration with the simulation environment.The hardware performance has been successfully verified.

Chapter 7

Learning and Experiences

1) Adaptability and Flexibility: The internship exposed me to real-world challenges and the need for adaptability in simulated and dynamic environments. Adjusting strategies, and finding alternative solutions taught the importance of flexibility in achieving project objectives.
2) Technical Skill Development: The internship provided a platform to develop technical skills in robotics, hardware integration, software development, and algorithmic design. The hands-on experience gained through practical implementation and problem-solving enhanced proficiency in these areas.
3) Effective Communication: Collaborating with team members and stakeholders throughout the internship emphasized the importance of effective communication. Presenting ideas, discussing challenges, and conveying progress and results in a clear and concise manner facilitated seamless teamwork and project coordination.
4) Professional Growth: The internship provided opportunities for personal and professional growth. It nurtured a sense of responsibility, accountability, and self-motivation. It also instilled a drive for continuous learning and improvement, laying a foundation for future endeavours in the field of robotics.

7.1 Key finding and insights

Through the project, several key findings and insights were obtained.

1 The use of a 3-degree-of-freedom planar robotic manipulator is suitable for docking applications, providing sufficient mobility and flexibility for precise positioning and alignment.

2 Simulation plays a crucial role in system development, allowing for thorough testing and validation before hardware implementation. It enables the identification of potential issues and provides a platform for algorithm development and optimization.

3 Motion planning and inverse kinematics algorithms are essential for generating smooth and collision-free trajectories. They ensure accurate positioning of the robot arm and facilitate successful docking operations.

4 Hardware implementation requires careful calibration and verification to align the physical robot with the simulated model. Close attention must be paid to mechanical tolerances, sensor accuracy, and calibration errors to achieve accurate and reliable performance. Overall, this internship significantly contributed to my technical and professional growth, equipping me with practical skills and experience in robotics, hardware integration, and system development.

Challenges Faced and Overcoming Them: Challenges associated with 3R planar manipulators for docking applications include workspace limitations, singularity avoidance, collision detection and avoidance, and real-time control. The workspace of the manipulator must be optimized to accommodate the docking requirements while considering the limitations imposed by the manipulator's kinematic structure. Singularity avoidance techniques are crucial to ensure stable and continuous motion during the docking process. Collision detection and avoidance algorithms are necessary to ensure the safety of the manipulator and the target object. Real-time control of the manipulator, considering the dynamic nature of the docking environment, poses additional challenges in terms of computational efficiency and system response time.

Chapter 8

Conclusion

In this project, a docking system using a 3-degree-of-freedom planar robotic manipulator was developed. By leveraging ROS, the MoveIt framework, and hardware interfaces, the project successfully demonstrated the capability to position the end effector accurately.The integration of Dynamixel motor-based hardware, MoveIt framework, and real-time path planning using the RRT Connect algorithm has yielded an impressive and efficient robotic arm system. Precise motor control has been achieved through the application of forward and inverse kinematics, enabling accurate positioning of the arm based on joint angles and end effector positions. In summary, this integration highlights the successful realization of a hardware model robotic arm with effective path planning, exemplifying the synergy of cutting-edge technologies in the field of robotics. The project successfully accomplished these objectives and achieved significant milestones in the following areas:
1. Simulation Environment: The ROS framework, along with RViz and Gazebo, was utilized to create a realistic simulation environment. The RRR robot model was visualized, and its behaviour and kinematics were studied, providing valuable insights into the system's dynamics.
2. Robot Design and Control: A URDF model was created, defining the structure, dimensions, and joint properties of the 3R planar manipulator. The MoveIt package was developed for motion planning and control, incorporating trajectory planning and inverse kinematics algorithms.
3. Hardware Implementation: The hardware components, including workspace

point identification, object position detection, camera calibration, and motors, were interfaced and tested. The movements of the physical robot arm were validated against the simulated results.

4. Performance Evaluation: The system's performance was evaluated by comparing simulated and hardware performance. The results demonstrated a high level of agreement, indicating accurate computation of joint angles and smooth motion execution. The achieved accuracy and precision in the docking process met the specified tolerances.

Chapter 9

References

1 http://wiki.ros.org/noetic/Installation/Ubuntu

2 http://wiki.ros.org/catkin/CMakeLists.txt

3 http://wiki.ros.org/catkin/package.xml

4 http://wiki.ros.org/ROS/Tutorials

5 https://www.rosroboticslearning.com/forward-kinematics

6 https://www.rosroboticslearning.com/inverse-kinematics

7 https://ros-planning.github.io/moveit_tutorials/